Distributed Systems

COMP90015 2017 Semester 1
Tutorial 02

Q1. Define and briefly explain each of the following distributed system challenges:

- Heterogeneity
- Openness
- Security
- Scalability
- Failure Handling
- Concurrency

Heterogeneity and mobile code

Mobile code is sent from one computer to the anther to run at the destination (e.g. Java applets):

- Code that is compiled to run in one OS does not run in another:
  - ♦ different hardware
  - ♦ different OS versions
- Virtual Machine approach provides a way of making code executable on any hardware - compiler produces code that is interpreted by the virtual machine
- Cross-platform compilation and code portability is another way, that compiles source code to multiple targets.

Q1. Define and briefly explain each of the following distributed system challenges:

- Heterogeneity
- **Openness**
- Security
- Scalability
- Failure Handling
- Concurrency

Openness

Openness refers to the ability of extend the system in different ways by adding hardware or software resources. Following are some approaches to address openness:

- Publishing key interfaces
- Allowing a uniform communication mechanism to communicate over the published interfaces
- Ensuring all implementations adhere to the published standards

Q1. Define and briefly explain each of the following distributed system challenges:

- Heterogeneity
- Openness
- Security
- Scalability
- Failure Handling
- Concurrency

Security

- There has three aspects of security:
    - Confidentiality (protection against disclosure to unauthorized individuals)
    - Integrity (protection against alteration and corruption)
    - Availability (protection against interference with the means of access)
- Security Mechanisms:
    - Encryption (e.g. Blowfish, RSA)
    - Authentication (e.g. passwords, public key authentication)
    - Authorization (e.g. access control lists)

Q1. Define and briefly explain each of the following distributed system challenges:

- Heterogeneity
- Openness
- Security
- **Scalability**
- Failure Handling
- Concurrency

Scalability

A system is considered to be scalable if it can handle the growth of the number of users. Scalability Challenges:

- Cost of physical resources: For a system with n users to be scalable, the quantity of physical resources required to support the system should be O(n) - if one file server can support 20 users, then two servers should be able to support 40 users.
- Controlling the performance loss: Complexity of algorithms that are used for searches, lookup etc should be scalable - for example a search algorithm for n data items requiring O(log(n)) search steps is scalable but one that requires n^2 is not.
- Resources should not run out: e.g. 32-bit Internet IP addresses running out requiring a new version of 128-bit address.
- Avoiding Performance bottlenecks: Decentralized architectures and algorithms can be used to avoid performance bottlenecks.

Q1. Define and briefly explain each of the following distributed system challenges:

- Heterogeneity
- Openness
- Security
- Scalability
- **Failure Handling**
- Concurrency

Failure Handling

Following are approaches to handing failures:

- **Detecting:** some types of failures can be detected, e.g. checksums can be used to detect corrupted data in a message, and some kinds of failure are hard to be certain about, e.g. the failure of a remote server.
- **Masking:** some failures that have been detected can be hidden or made less severe, e.g. timeout and message retransmission.
- **Tolerating:** it is sometimes impractical to try and handle every failure that occurs, sometimes it is better to tolerate them, e.g. failure is reported back to the user, e.g. web server not being available, try again later, or video is rendered with errors
- **Recovery:** failure sometimes leads to corrupted data and software can be designed so that it can recover the original state after failure, e.g. implementing a roll back mechanism.
- **Redundancy:** services can be made to tolerate failures using redundant components, e.g. multiple servers that provide the same service, as so called fail over.

Q1. Define and briefly explain each of the following distributed system challenges:

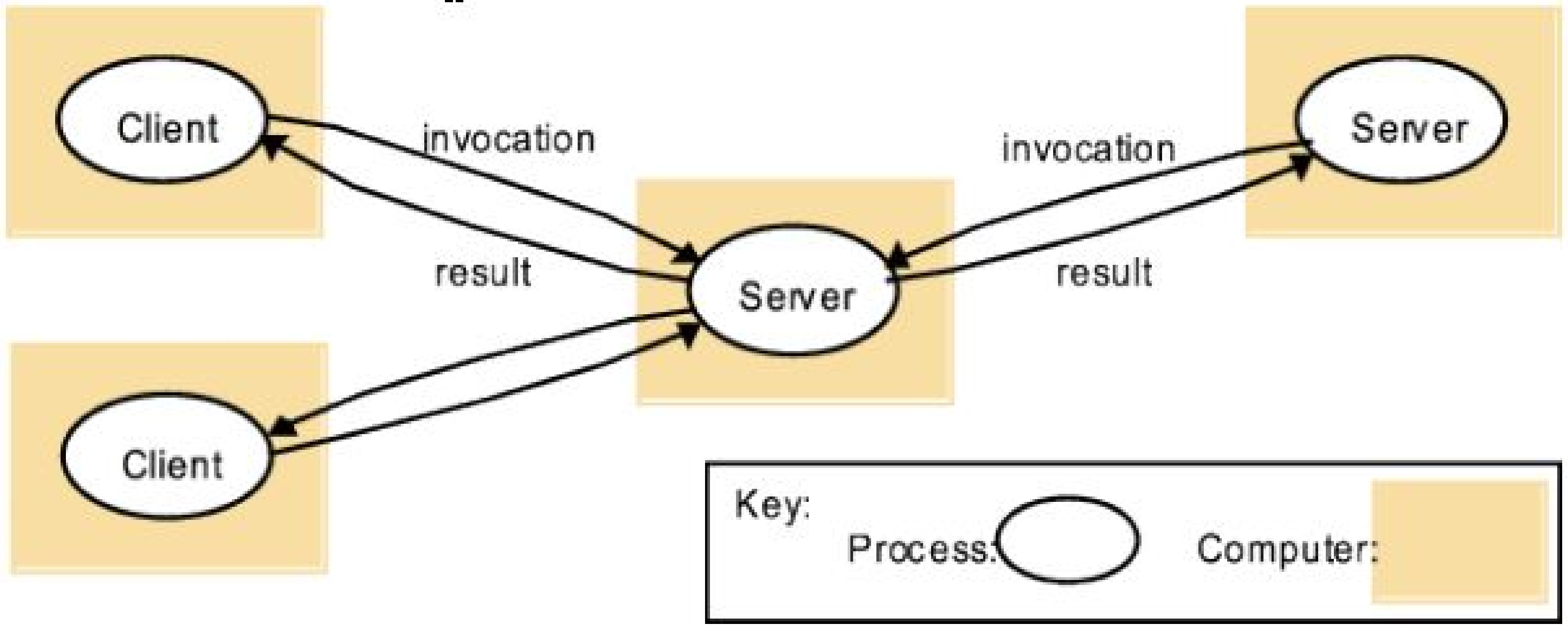
- Heterogeneity
- Openness
- Security
- Scalability
- Failure Handling
- **Concurrency**

Concurrency

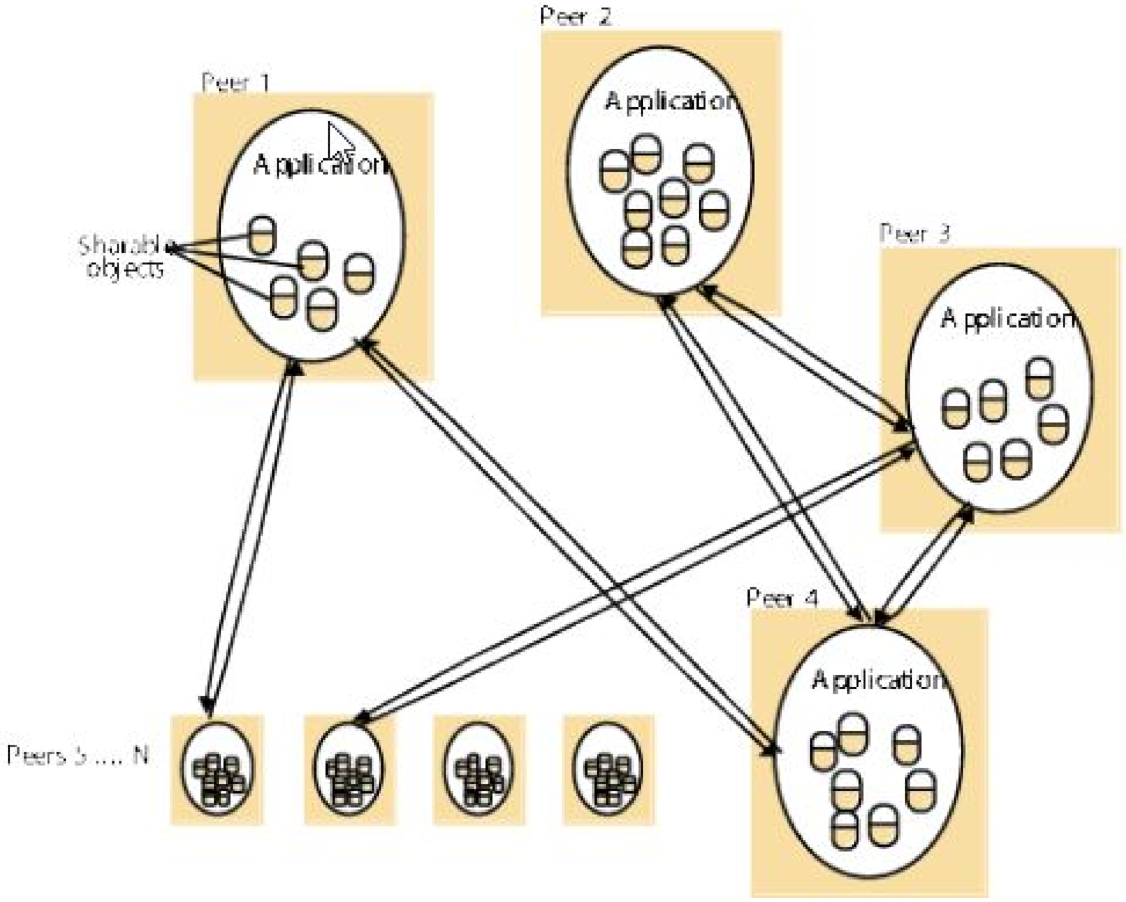
- Multiple clients can access the same resource at the same time, in some cases for updates
- One approach to handling concurrency is making access sequential - slows down the system
- Semaphores supported by the operating system is a well accepted mechanism to handle concurrency

Q2. Briefly explain the difference between a client-server architecture and a peer-to-peer architecture.

Client-server



Peer-to-Peer

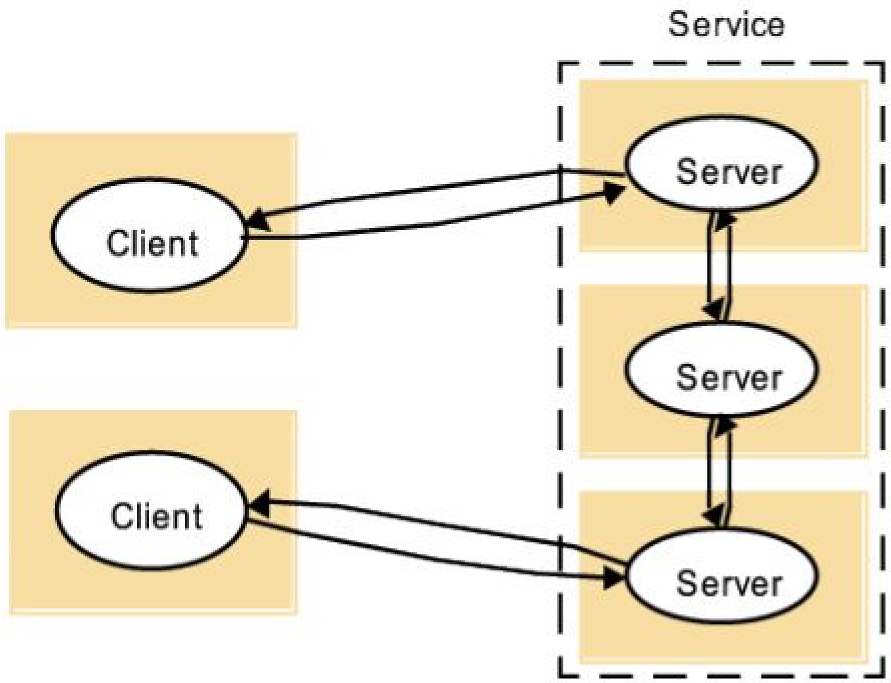

Peer 2
Application
Peer 1
Application
Sharable objects
Peer 3
Application
Peer 4
Application
Peers 5 .... N

Q3. Briefly explain each of the following distributed system architecture variations, giving also a reason or a benefit its use:

- Services provided by multiple servers
- Proxy servers and caches
- Mobile code and Mobile Agents
- Network computers
- Thin clients
- Tiered Architecture

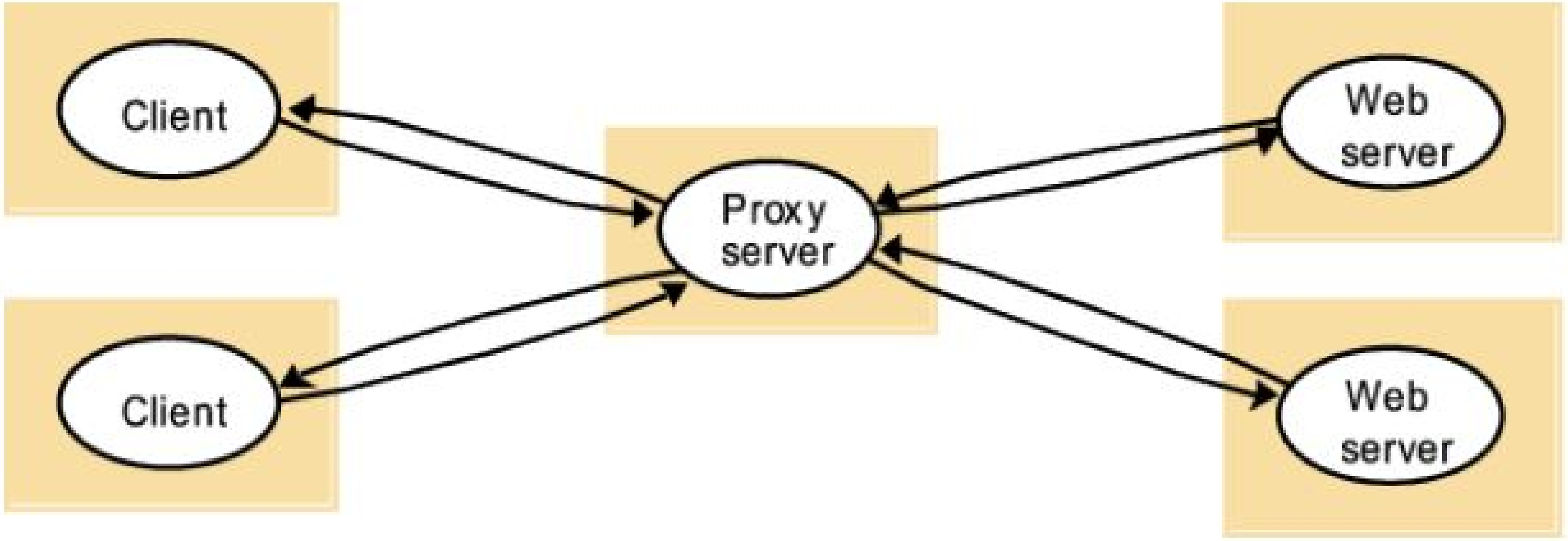
A service provided by multiple servers

Service is provided by several server processes interacting with each other. Objects may be partitioned (e.g web servers) or replicated across servers (e.g. Sun Network Information Service (NIS)).



Proxy servers and caches

- Cache is a store of recently used objects that is closer to client
- New objects are added to the cache replacing existing objects
- When an object is requested, the caching service is checked to see if an up-to-date copy is available (fetched in not available)



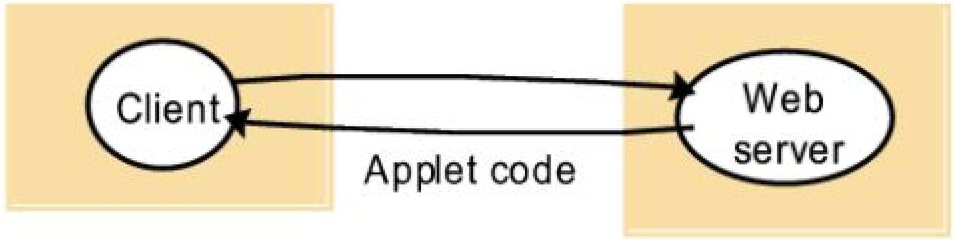
Mobile Code and Agents

Mobile Code is down loaded to the client and is executed on the client (e.g. applet).

Mobile agents are running programs that includes both code and data that travels from one computer to another.

E.g. Web Applets:

a) client request results in the downloading of applet code



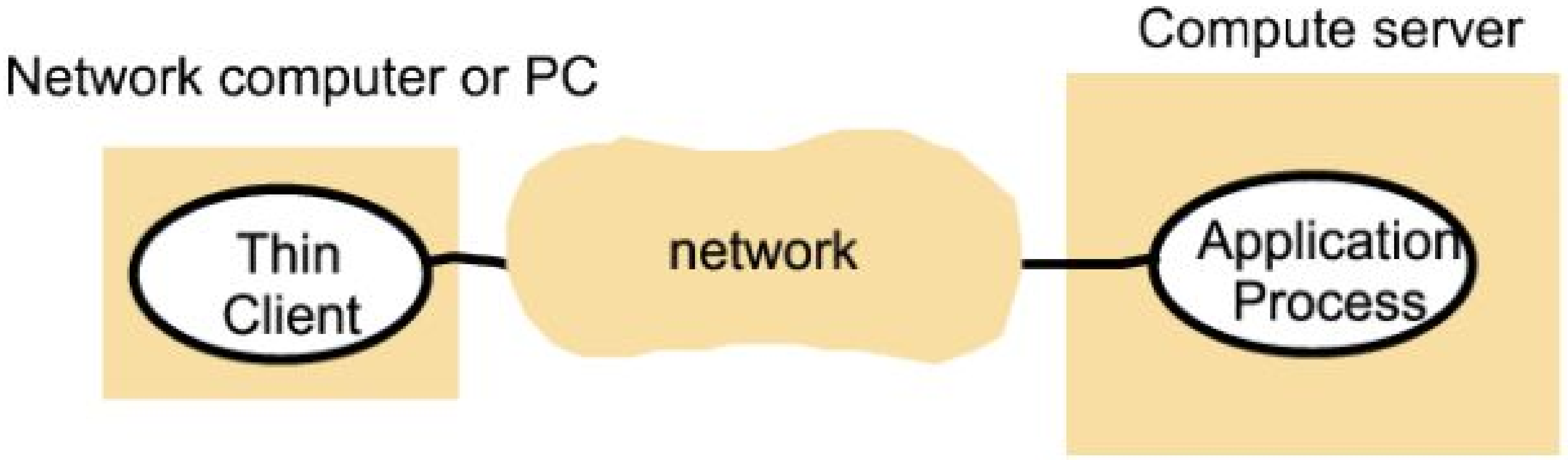
b) client interacts with the applet

Client
Applet
Web server

Network Computers and Thin clients

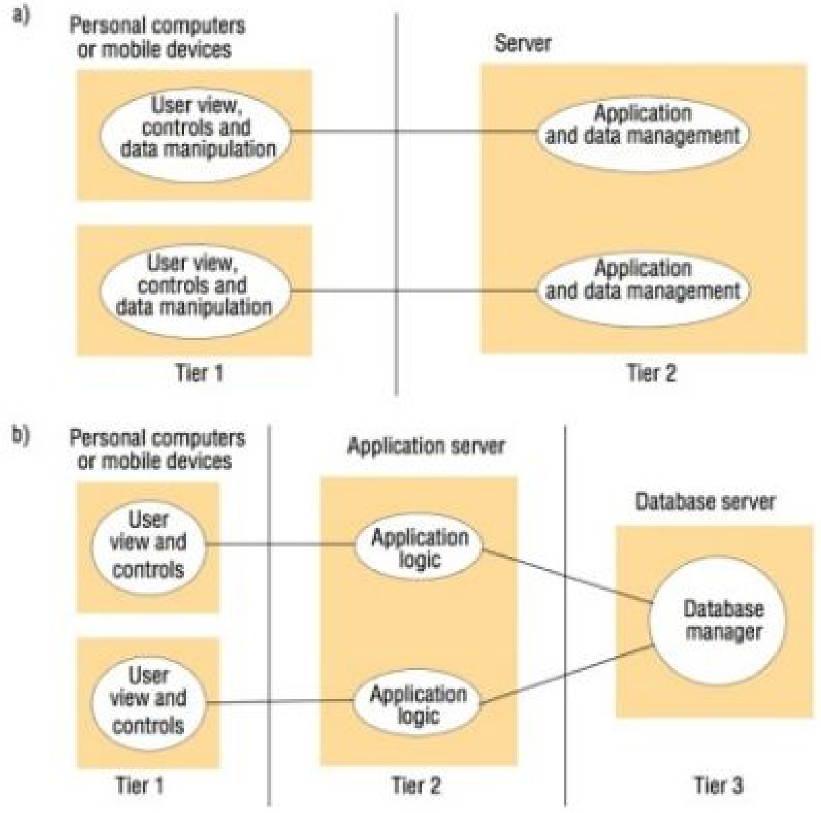
- **Network Computers**: download their operating system and application software from a remote file system. Applications are run locally.
- **Thin Clients**: application software is not downloaded but runs on the computer server - e.g. UNIX.

This paradigm is usually not suitable for highly interactive graphical activities.



Tiered architecture

Tiered architectures are complementary to layering. Layering deals with vertical organization of services.


a)
Personal computers or mobile devices
Server
User view, controls and data manipulation
Application and data management
User view, controls and data manipulation
Application and data management
Tier 1
Tier 2
b)
Personal computers or mobile devices
Application server
Database server
User view and controls
Application logic
Database manager
User view and controls
Application logic
Tier 1
Tier 2
Tier 3